Simultaneous Task Subdivision and Allocation using Negotiations in Multi-Robot Systems

Claudio Rossi¹*, Leyre Aldama¹ and Antonio Barrientos¹

1 Centre for Robotics and Automation UPM-CSIC, Madrid, Spain
* Corresponding author(s) E-mail: claudio.rossi@upm.es
Received 28 May 2014; Accepted 27 October 2014
DOI: 10.5772/59880
© 2015 The Author(s). Licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract
This paper presents a negotiations-based approach for simultaneous task subdivision and assignment in heterogeneous multi-robot systems. We first propose an abstraction of the concept of a task that allows for the generalizing of a variety of different problems. Based on such an abstraction, we have developed a negotiation protocol based on Rubinstein’s alternate offers protocol. This is extended to the multi-dimensional space and employs a heuristic search step for evaluating and generating offers. Furthermore, the issue of how to extend a bilateral negotiations protocol to more than two parties is taken into consideration. The protocol was first tested in numerical simulations with different scenarios and then applied to three real-world missions.

Keywords Multi-robot systems, Robot cooperation, Negotiations, Bargaining, Task partition and allocation

1. Introduction
Multi-robot systems (MRSs) represent a very active field of research. A variety of techniques have been proposed in order to approach the problem of coordination in different kinds of applications [19, 33]. Cooperation applications can be roughly divided into two classes: tight cooperation requires continuous coordination between the robots such as, for instance, in box pushing and formation keeping. Loose cooperation requires coordination at the beginning of the mission for planning the division of labour and at given points in time when re-planning may be needed. Exploration, mapping and surveillance are typical applications. Behaviour-based [18, 32], schemas [1] and virtual vector fields [14, 29, 20] are examples of techniques suitable for the first class of coordination problems, while market-based [9, 8], auction [11] and bargaining [13] techniques are commonly used for the second class of problems. Multi-robot coordination techniques, like virtual potential fields,

¹ In this paper, we will use the terms ‘robot’, ‘vehicle’, ‘negotiator’, ‘agent’, ‘party’ and ‘team-member’ synonymously, as this makes little difference for the scope of the discussion. To be precise, a vehicle is a mobile robot that runs a negotiator agent, besides other control software, that is a party in a negotiation process within the team of robots of which it is a member.
are also used in sensor networks applications (see, e.g., [16]) for coordinating communication tasks or energy management for teams of mobile sensors.

Here, we focus on loose cooperation for teams of mobile robots. In this class of problems, a given task has to be partitioned into sub-tasks, and the sub-tasks have to be assigned to individual team-members for execution. Most of the coordination techniques assume that the task subdivision step is done at a high level, either on a central station or by a team leader, and as such focus on the sub-task allocation problem.

This approach, although applied with success in many applications, has two principal drawbacks: first, it is centralized (task partitioning is done in one place), and second the partitioning algorithm is usually considered “outside” the coordination protocol. Often, the details of how the original task is partitioned are not given at all.

As pointed out in [34], “the main drawback of this approach is that task decomposition is performed without knowledge of the eventual task allocation; therefore, the cost of the final plan cannot be fully considered. Since there is no backtracking, costly mistakes in the central decompositions cannot be rectified.”

One of the most popular protocols for task assignment is the contract net [26], and many algorithms [12, 27, 5, 9, 11, 24, 15] have been based on this. In all these approaches, each robot has predefined cost and revenue functions that are used to compute the expected gains and losses for performing tasks. However, the robots’ preferences and limitations are considered only in the assignment stage, when the robots decide whether to accept (or opt for) a task or not.

Such features do not suit our need for a fully distributed approach that should consider the robots’ capabilities early on at the task partitioning stage. A task partitioning algorithm which does not take these into account may produce solutions that are not feasible, leading to incomplete task execution since no agent will bid for certain task(s).

The negotiation protocol we propose performs a simultaneous task subdivision and allocation in a distributed way, taking into account the robots’ preferences at all times.

Negotiations have been widely studied in the context of socio-economic studies [7] using, among others, game theory [17], and have been applied, e.g., to electronic commerce using agents [28]. The main problem with game theory approaches is that their theoretical results often refer to simplified models that are not immediately applicable to complex applications. The protocol we propose is based of Rubinstein’s alternate-offers protocol [22]. Since such a protocol is based on a uni-dimensional good, we first had to develop an extension to the multi-dimensional space. In this context, a search mechanism for the best (counter)-offer had to be devised for the protocol to be applied. Furthermore, Rubinstein’s protocol has been developed for bilateral negotiations. The issue of how to extend it to more than two parties will be discussed below.

It must be pointed out that in all market-based approaches, the task partitioning and assignment is performed taking into account the status and knowledge of the world available at the time of the assignment. An instantaneous assignment [10] (i.e., static) is performed. Any situation not known at the time of negotiation will thus not be taken into account. The work presented here belongs to this class of approaches, and thus shares such limitations. However, renegotiations can deal with most of such situations, e.g., a robot failure or an obstacle that does not allow a team member to accomplish the task assigned. For instance, in [34] an extension is proposed where auctions can be initiated by any agent to (re-)allocate sub-tasks at any time, achieving dynamical allocation that can improve the solution at any reallocation. Here, we foresee a similar approach, were renegotiations can happen in case of need. Task/mission advance monitoring, which would trigger a renegotiation, is beyond the scope of this paper.

2. Tasks

In order to design a negotiation algorithm that is general enough to work with different kind of tasks (cf. Fig. 1), an abstract task concept should be defined. A negotiation algorithm based on such an abstraction allows different applications with minimal changes.

Figure 1. Examples of the instantiation of tasks: areas, a set of target points, a communication chain and a frontier. All the tasks can be defined by an array of real numbers: (a) coordinates of the vertices of the polygons $(x_r, y_r)$; (b) coordinates $(x_r, y_r)$ and values of the targets $(t)$; (c) points in space $(x_r, y_r, z_r)$ and communication ranges $(r, \gamma)$; (d) curve parameters (curve centre $(x_c, y_c)$, radius $r_f$, and limit angles $(\gamma, \delta_f)$).

In the context of loose cooperation, we focus on the object to be divided and not on the activity to be performed on
such an object. For example, if the task is surveying a given area, we are mainly interested in partitioning the area and assigning sub-areas to the agents. The activity the agents will have to perform and their preferences (for instance, with regard to their capabilities) will of course play an important role in the negotiation. This role is encapsulated in the cost/reward the agents associate with the task, as explained later on.

We define a task $T$ as an element of a set $\mathcal{T}$, $T \in \mathcal{T}$. An element of $\mathcal{T}$ is described by a set of $k$ parameters:

$$x \in \mathbb{P}^1 \times \ldots \times \mathbb{P}^h \ni \sum_{i=1}^h k_i = k,$$

$\mathbb{P}^i$ for $i = 1 \ldots h$, being parameter-types and $k_i$ their respective numbers. In general, parameters can be mixes of any types. For simplicity and without loss of generality, we can assume that they are all of the same type $\mathbb{P}$.

We consider $T(x) = \tau(x)$, i.e., a task $T$ is the result of a function that maps a set of parameters onto a task: $\tau : \mathbb{P} \rightarrow \mathcal{T}$.

The problem is to divide a task $T$ into a set of $R$ sub-tasks: $T(x) = [T_1, \ldots, T_R]$. Each sub-task $T_i$, $i = 1 \ldots R$, can in turn be described by a set of parameters $x_i$, where sub-tasks $x_i$ can have different numbers of parameters. Then:

$$T(x) = \{ T_1(x_1), \ldots, T_R(x_R) \}.$$

In general, a good subdivision is such that there is minimum overlap between sub-tasks (ideally null), and such that the union of the sub-tasks cover the original task. That is:

$$T_i \cap T_j = \emptyset, \quad \forall i, j = 1 \ldots R, i \neq j \quad \text{and} \quad \bigcup_{i=1}^R T_i = T,$$

where the operators $\cap, \cup : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$ (overlapping and union) are to be defined according to the meaning of the task.

Note that there can be exceptions, depending upon the application. For example, in a communication relay application, the overlapping between the communication ranges of two robots must not be null (See Fig. 1(c)).

Let $g : \mathcal{T} \rightarrow \mathbb{R}$ be a reward function, giving the value of a (sub-)task. Then, the function:

$$f : \mathbb{P}^k \rightarrow \mathbb{R} = \tau \circ g$$

associates a reward with a set of parameters describing a task. We associate with a subdivision $T_0 = [T_{ij} \ldots T_k]$ an index called ‘global coverage’ $G$ which takes into account the total coverage of the sub-tasks and their pair-wise overlapping.

$$G = \sum_{i,j=1}^R f(x_{ij}) - \frac{1}{2} \sum_{i=1}^R \sum_{j=1}^R g(T(x_i) \cap T(x_j))$$

As such, the problem of task subdivision can be formulated in the following way: given a task $T_0$ and a number $R$ of agents, find the $R$ sets of parameters $x_i$, $i = 1 \ldots R$, such that $G$ is maximized:

$$\max_{x_1 \ldots x_R} (G).$$

$G$ is a global performance index that is used in order to evaluate a solution a posteriori. During the negotiation, each robot can give a different value to the same task, depending upon its characteristics (locomotion, sensors, status, etc.). In other words, each robot uses its own reward function $g_i$ to evaluate a task.

Since each agent looks for its own reward, there is no guarantee that index $G$ corresponds to a solution that is optimal. However, as we will see in the next sections, good sub-optimal values are usually achieved.

2.1 Evaluation of a task

During the negotiation, each robot has to evaluate the cost and reward of a given task. To this aim, it takes into account its internal parameters to evaluate the cost of executing the task, the start-up cost (e.g., to reach the execution site), specific constraints (e.g., forbidden zones, turn angles, battery level) and the general reward associated with the task, expressed as the function $g$.

Thus, given the complete task $T_0$, the evaluation function $g_i$ of a sub-task $S_i$ for an agent $i$ takes the form of a weighted sum of terms, such as the dimension of the task (e.g., area, length, number of targets), the distance from the initial position of the robot to the task execution starting point, the penalties for overlapping sub-tasks and for the part of $S_i$ outside $T_i$ (ct. Table 1). These last two factors are important, as they balance the importance of avoiding overlapping and not going outside mission limits. They can be easily visualized in the context of area partitioning: partial overlapping may be allowed (or even desired). In the same way, going outside the assigned area can be permitted (e.g.,

---

3 In most practical cases, $x$ will be an array of real numbers: $x \in \mathbb{R}^k$.
4 Higher orders which overlap (i.e., the overlapping of three or more tasks) are not considered since their size decreases very quickly during the negotiation process, and thus their contribution is not significant.
for an aeroplane to make a u-turn) according to the constraints of the mission.

\[ g(S_i) \text{ Value of the task} \]
\[ \text{dim}(S_i) \text{ Size} \]
\[ \text{dist}(\text{pos}, \text{site}) \text{ Distance to task initial point} \]
\[ \text{dim}(S_i \setminus T_j) \text{ Part of the task exceeding the target task} \]
\[ \forall \text{sub}_i, \text{dim}(S_i \cap S_j) \text{ Tasks overlapping} \]

Table 1. Example of factors taken into account when evaluating a (sub-)task \( S_i \)

2.2 Example

To illustrate the concept of task evaluation, consider the case where a ground vehicle (Robot 1) and an air vehicle (Robot 2) have to explore a given area \( T_0 \) where a no-fly zone NFZ is present (see Fig. 2). In this case, the function \( g \) is simply the area of the region and each vehicle will try to maximize its share. However, the constraint on the NFZ will cause Robot 2 to evaluate each region (sub-task) containing it negatively. The reward functions are defined as:

\[ g_i(S_i) = \alpha \cdot \text{dim}(S_i) - \beta \cdot \text{dim}(S_i \setminus T_j) - \gamma \cdot \sum_{i \neq j} g(S_i \cap S_j) - \phi \cdot C_i \]

\[ i = 1, 2 \quad C_1 = 0, \quad C_2 = \text{dim}(S_2 \cap \text{NFZ}) \]

where \( C_2 \) encapsulates Robot 2’s particular constraints.

The constants \( \alpha = 1, \beta = 1, \gamma = 1 \) and \( \phi = 1000 \) have been set in such a way that it is equally penalized going outside of the target area and overlapping, and Robot 2 strongly penalizes having to explore region \( S_2^\frac{1}{2} \). In this example, \( g_1(S_1) = 0.26, \quad g_1(S_2) = 0.41, \quad g_2(S_1) = -119.74, \quad g_2(S_2) = 0.41 \). Robot 2 will not accept subdivisions that include the forbidden zone because its revenue will be negative. Note, however, that in any case \( G = 0.259 + 0.409 + 0.668 \), a value close to the optimum (0.673). Note that a partitioning of the task with minimal overlapping and optimal coverage alone does not guarantee a feasible solution: only if this assigns the NFZ to Robot 1 will the mission be successfully carried out.

3. Task negotiations

Given a team of \( R \) robots and a task \( T_0 \), the purpose of the negotiation process is to find a suitable subdivision of the task into \( R \) sub-tasks, and an assignment of each sub-task to a robot. Our aim is to perform these two actions simultaneously and in a distributed way. In our system, the number of sub-tasks is determined by the number of robots willing to participate in the negotiation. We assume that robots are willing to perform as much as they can of a given task (hence maximizing their reward), the only limitations being their available resources (e.g., endurance, computational power, battery consumption, etc.). In a negotiation, each agent will try to maximize its reward by: (i) trying to get the biggest possible sub-task, and (ii) minimizing overlapping other agents’ tasks. Thus, each agent starts proposing the biggest possible share for itself, and reduces it until its counterpart finds it acceptable. In this way, a good near-optimal solution can be achieved and the global index \( G \) is optimized in a distributed way, without even being computed explicitly.

3.1 Alternates offers

In the alternate-offers protocol proposed by Rubinstein [22], each part of the bilateral negotiation, in turn, proposes the share \( p \) it would like to receive of a uni-dimensional good of size 1. The responder will clearly get \( 1 - p \). The responder can agree to such a subdivision or else disagree to it, in which case it has to propose a counteroffer. The protocol assumes that each party has a negotiation cost called a discount factor, by which it decrease its offers at each round, starting by claiming the whole good. Such a protocol has interesting theoretical properties: it guarantees a termination and can forecast the final agreement, which will be a perfect equilibrium in the sense of game theory.

The update rule of a target share \( p \) is:

\[ p_{t+1} = p_t \cdot \delta^t, \quad t = 0 \ldots n, \quad p_0 = 1, \quad \delta < 1. \] (1)

In this example and in the following ones, a linear combination of factors is considered. However, any kind of function can be designed, e.g., taking into account non-linearities of the robot’s performance or of the reward associated with a task (e.g., exponential penalties).

Figure 2. Negotiation with a forbidden area (blue area). The green agent (Robot 2) is not allowed to fly over the buildings, and will refuse all proposals, including the no-fly zone. In this example, \( \text{dim}(T_0) = 0.673 \), \( \text{dim}(\text{NFZ}) = 0.03 \), \( \text{dim}(S_1) = 0.409 \), \( \text{dim}(S_2) = 0.259 \), \( \text{dim}(S_1 \cap S_2) = 0.0 \), \( \text{dim}(S_1 \setminus T_0) = 0.0 \) and \( \text{dim}(S_2 \setminus T_0) = 0.0 \).
Let the discount factors be $\delta_1$ for the initiator and $\delta_2$ for the responder. Rubinstein’s theory guarantees that one perfect equilibrium point exists such that the share that $p_{\text{final}}$ the initiator agent, will receive is:

$$p_{\text{final}} = \frac{1 - \delta_2}{1 - \delta_1 \delta_2}. \quad (2)$$

If the discount factors were known to both parties, each could know without negotiating what will be its share. We will discuss this point later in this section. Note that smaller values of $\delta$ mean smaller shares: if an agent is more impatient to reach an agreement (at each step, its counteroffer is reduced more than its opponent’s), it will receive a smaller part of the good.

This protocol is not immediately applicable to the multi-dimensional case. In the uni-dimensional case, when an agent makes a proposal $p$ of what it would like to receive, it is immediate that the other will receive $1-p$. In the multidimensional case, given a proposal $x \in \mathbb{R}$ on the whole task $T_p$, an agent will receive $T_p \setminus T(x)$, the part of $T_p$ not covered by $T(x)$. However, the evaluation of such a portion is not always straightforward. For example, in the case of areas, one would prefer that the region it has to explore be connected and well-shaped, and in the case of foraging, an optimal path connecting all the targets should be computed in order to determine the actual cost of reaching all the targets. Thus, an agent will perform some kind of search in the space $T_p \setminus T(x)$ to decide if the share it would receive is acceptable. In the same way, when it comes to updating a previous offer that has been rejected, many different configurations are possible since any of the parameters can be changed, and this can produce many different new proposals that must be taken into account in order to decide which is the most convenient. In practice, this means searching the space for a solution that minimizes the overlapping while maximizing the share.

### 3.2 Searching for offers

We divide the negotiation into two levels: the protocol level and the proposals’ evaluation and generation level. The protocol level is governed by such parameters as impatience to reach an agreement (time pressure as a discount factor $\delta$) and the desired target reward. This level basically implements Rubinstein’s alternating offers protocol, also managing the initiation and termination of the negotiation.

The proposal generation level searches the space for a good share given a proposal from the other party, taking into account its own characteristics. This level is also responsible for updating counteroffers.

Both the proposal evaluation and update search steps can be performed by a variety of algorithms. We have experimented with evolutionary algorithms in those cases where a complex space has to be searched (e.g., in the case of areas) and with complete discrete search algorithms ($A^*$ [23]) for the simplest cases, such as the negotiation of target points (see the next sections).

Figure 3 shows the communications protocol between two agents. The starter agent start the negotiation, sending a request to its team-mates (only one is depicted). In case the responder agrees to participate, a bargain loop is entered into where the two parties alternately receive an offer, evaluate it and decide whether to accept it or not, in the latter case formulating a counteroffer.

![Diagram](image)

**Figure 3.** Negotiation loop. The loop can be interrupted at any moment due to an abort message of one part or a time-out when waiting for an offer (not shown in the sequence diagram for clarity). The negotiation loop can also be interrupted at any moment if one of the parties decides to unilaterally abort the negotiation, in which case a time-out occurs while waiting for a message, or if the mission is interrupted by a command and control station.

#### 3.2.1 Termination

As mentioned earlier, if the discount factors were known, each agent could compute in advance the maximum share it would receive. Such factors are private information of the robots, but they can be estimated by comparison of the offers received. With these, it can estimate the maximum possible share it can expect, according to Rubinstein’s theory.

Supposing that Agent 1 is the first to make a proposal, then it can estimate $\delta_2$ and forecast that its share will be:

$$\hat{p}_1 = \frac{1 - \hat{\delta}_2}{1 - \hat{\delta}_1 \hat{\delta}_2}. \quad (3)$$

where $\hat{\delta}_2$ is the estimated discount factor of the opponent. Such an estimation is performed by taking into account the
sizes of the received offers over time, the simplest estimation being the ratio of the two last proposals received, i.e., \( \frac{\text{dim}(T^0)}{\text{dim}(x^{(t-1)})} \). More complex statistical techniques can be used according to need (e.g., averaging the ratio over the last \( n \) steps).

The values \( \hat{p}_1 \) and - symmetrically - \( \hat{p}_2 \) can be used by the agents as a reference term to decide whether to accept an offer (and terminate the negotiation) or not.

In practice, if the evaluation of the share provides a value sufficiently close to \( \hat{p} \), the negotiator knows that the theoretical limit (equilibrium point) has been reached, and thus that the negotiation has reached its end. Thus, in order to comply with Rubinstein’s theory’s hypotheses, the robots’ new offers for a task at time \( t \), \( T^0 \), should have a dimension \( \text{dim}(T^0) = \delta \cdot \text{dim}(T^{(t-1)}) \), where \( \delta \) should be constant. In some applications, however, such constraints cannot be guaranteed. In this case, the proposed protocol can still be used, although the termination criteria, of course, lose their mathematical foundations and the value \( \hat{p} \) should be considered as an approximation and taken into account accordingly.

When an agreement has been reached, the result is a subdivision of the original task and - at the same time - an assignment of the sub-tasks. Note that in this way one agent does not need to possess information about the private characteristics of its team-mates. The only information it needs concerns their offers, in the form of an array of parameters \( x \in \mathbb{P}^k \).

### 3.2.2 Learning

In our negotiation strategy, two learning processes take place. Explicit learning is performed by estimating the opponent’s discount factor, as described above.

Additionally, each time a new offer is received, the search in the parameters’ space for the generation of the counter-offer is not started from scratch but rather takes as its starting point the previous offer and the response received to this. Therefore, the search is directed towards promising regions of the parameters’ search space (and away from bad ones). In other words, the search algorithm adapts its search according to the responses of the other agent. In this sense, we can say that the search algorithm learns the opponent’s preferences and constraints, and produces offers that implicitly depend upon them.

The NFZ application described earlier is an example. One agent is constrained to avoid certain parameters’ combinations that define a forbidden area. Although the other agent is not aware of such constraints, as they are private information, all offers that include the forbidden configurations will be rejected and so the other agent will be forced to generate offers that avoid them.

#### 3.2.3 Analysis

To illustrate the behaviour of a negotiation, Figure 4 shows an example of the shares of area partitioning between two agents. The left plot shows how the shares the agents obtain is actually close to values predicted by Rubinstein’s theory. In the numerical simulation, the negotiation ends when one agent estimates that the share it is receiving is greater than or equal to its prediction, which is made by estimating the opponent’s discount factor (Eq. 3).

The plot to the right shows how the overlapping task is reduced during the negotiation, and how the global coverage \( G \) is very close to the optimum value (in this case, the total area \( T \)). Moreover, note how good global coverage \( G \) is maintained throughout the negotiation. In other words, the whole area is covered by the two robots at all times.

![Figure 4](image)

**Figure 4.** Negotiation on how to partition an area. Best share and Rubinstein predictions (left). Total area \( T \) and global coverage \( G \) (right). \( \delta_1 = 0.95 \), \( \delta_2 = 0.955 \); both agents start claiming the whole area of size 0.74.

### 3.3 Extending to more than two negotiators

It is important to notice that, as pointed out in [6], “the difference between a two-party and a three-party game reflects a basic qualitative difference between the types of processes that take place within the negotiation. The involvement of any more than three parties can be seen as an extension of a three-party process.” In other words,
is a shift in complexity when passing from two to three negotiators, but passing from three to any number greater than three does not imply any difference in the negotiation process. Hence, in the following analysis we will focus the discussion on the case where \( R = 3 \).

When there are more than two agents, the proposed negotiation protocol can be extended in several ways.

Some of the heuristic algorithms that we have considered are:

- **Rounds**. Each agent, in turn, makes a proposal for its desired share and passes this on to a team-mate. At the end of the round, if all the agents are satisfied, the negotiation is closed, otherwise another round starts. In order to avoid privileging one individual, the turn to start the round passes on to a team-mate. For simplicity, the order is currently fixed by the agent’s number.\(^6\)

- **Coalitions**. The problem is reduced to a set of bilateral bargains. The agents are divided into two groups that negotiate a first subdivision. Next, the process is repeated inside each group, recursively.

- **All-against-all**. Each robot bargains bilaterally with all the others in order to reduce any overlap with each of them.

- **Unanimity**. In turn, each robot proposes a complete subdivision for all the robots. If there is full agreement, the negotiation ends, otherwise another round begins.

- **Unanimity with an exit option**. As with unanimity, each robot proposes a complete subdivision of the task. If a single robot (other than the proposer) agrees with the proposed subdivision, it takes its share and leaves the negotiation. Next, another round starts for sharing the remaining task among the remaining robots.

The last two strategies imply that each agent resolves the complete problem. They are difficult to implement as they would take excessive computational power, since the complete state of the system must be taken into account. Moreover, the private information of the other team-mates would be difficult to integrate. The coalitions strategy has the drawback of a mechanism for forming the two sub-teams by electing a representative for negotiating. This is far from trivial (see, e.g., [31, 25]). The number of bilateral negotiations would be of the order of \( \log_2 R \). The all-against-all strategy is easy to implement, but it has the drawback of computational and communication requirements, since in total there will be of the order of \( R^2 \) open negotiations.

Thus, the rounds strategy seems to be the best, both for simplicity of implementation and in terms of computational requirements. It is fully distributed and decentralized, it does not need complete system status information or team-mates’ private information, and the number of negotiations to be carried out is of the order of \( R \). Additionally, an exit option can be implemented such that an agent can accept an offer and leave the negotiation, while the others can keep on negotiating, e.g., in order to reduce overlapping between the respective desired shares.

### 3.3.1 Estimating the shares when \( R > 2 \)

In the “rounds” negotiation strategy, agent \( i \) has to consider the union of the offers received \( \bigcup_i \{ \delta(s) \} \) and estimate its discount factor \( \delta \). The factor by which the union of the offers received shrinks, \( \delta \), is not constant due to the overlapping of the proposals of the team-mates. In general, such overlapping is bigger in the beginning and gets smaller as the negotiation proceeds. Therefore, the estimation of the factor \( \delta \) will be bigger in the beginning (the union shrinks more slowly). However, the estimation refines and tends to the real value as the negotiation proceeds. In turn, the value of the expected share \( p \) will be smaller in the beginning and will then exhibit a growing trend, tending towards the actual value.

Even if, from the theoretical point of view, we are violating the requirement of constant discount factors, on the practical side the estimated values of \( \delta \) can then actually be used in Equation 3 in order to have an estimation of the expected share, and thus help in deciding when to stop the negotiation.

### 3.4 Communication and Computational Costs

A final remark must be made regarding the communications topology implied by the algorithm. In this work, we have assumed that all-to-all communications are available in order for the negotiation to take place. All-to-all communications are not strictly needed, since in the negotiation rounds it is sufficient to establish a communication ring. Moreover, communications will be available only during the negotiation itself, and are not required to be maintained continuously during the execution of the tasks that the robots have agreed on.

We also considered no bandwidth restrictions, as the amount of information exchanged is not a critical issue for our protocol. As an example, a polygon of 10 points is described by 20 floating point numbers. Assuming four bytes per number, a total of 80 bytes (plus headers) are sent with each offer. Taking into account the number of agents and the typical number of negotiation rounds needed, a total amount of the order of kilobytes for the whole process can be calculated.

Computational costs are centred on the proposal evaluation and counter-proposal update steps. Such costs depend on the application at hand and on the degree of precision needed. For instance, in an area survey application, in some cases the computation of a route may be required that

---

\(^6\) As suggested by an anonymous reviewer, an optimal ordering may optimize the number of rounds needed and/or the rewards. Modifications of the protocol and of the information exchanged will be needed for this purpose.
carefully takes into account the robots’ and the environment’s characteristics, while in other cases an approximation that simply takes into account the dimension of the areas and the reasons why the polygons’ intersections and unions can be employed, simplifying the evaluation. In any case, such costs are independent of the negotiation protocol used, since an evaluation step is always necessary to assess the goodness of a given task partition and assignment. Moreover, we would underline that, in a distributed approach such as the one that we propose, each agent only performs such computations for its sub-task, while in approaches where agents need to compute a complete solution, this cost must be multiplied by the total number of sub-tasks of the mission.

4. Numerical simulations

In this section, we present the simulations we have performed on two examples of tasks: areas and target points. In all the tests, the total size of the task is normalized to one and the discount factors refer to the percentage of the task the robots will reduce at each round. For example, a discount factor \( \delta = 0.99 \) means that for the next round the task claimed will be 1% smaller. Clearly, when the dimension of the task is a discrete quantity, as in the case of the target points, the agents will reduce the claimed share by one quantum every few rounds, when the claimed share reaches the next integer value.

4.1 Areas

A variable number of agents had to negotiate how to partition a given region described by a polygon. The number of points defining the polygon was set to six and included in a square of a normalized size of \( 1 \times 1 \). As such, \( k = 12 \), \( \mathbb{P} = \mathbb{R}^{12} \) and \( x = \{ p_1, \ldots , p_k \} = \{(x_1,y_1), \ldots , (x_n,y_n)\} \). The task evaluation function used is:

\[
g(x) = g(S) - \dim(S) - 2 \cdot \sum_{i<j} g(S \cap S_j) - 0.2 \cdot \text{ill}(S)
\]

where \( g(S) = \dim(S) \) is the area of the polygon described by \( S \). In order to penalize poorly-formed polygons, a penalty function \( \text{ill}(S) = \text{perimeter}(S) / \text{area}(S) \) was adopted. Table 2 reports the results of the simulations. For all the tests, the agents used the same discount factor \( \delta = 0.98 \).

As expected, more agents need more rounds to find agreement. As far as the quality of the solution is concerned, it must be pointed out that the area was not 100% covered, but the overlap between the sub-tasks is extremely low. Likewise, the part of the sub-task exceeding the target polygon \( T_0 \) is negligible (less then 1% in all cases). As explained earlier, such values can be modified with the fine-tuning of the weights used in the task evaluation function and according to the mission’s needs.

4.2 Target locations

In this case, the agents had to negotiate how to partition a given set of target locations. The task is then the set of coordinates of the points. The task evaluation function used is:

\[
g(x) = g(S) - \dim\left( \bigcup_{i<j} S_i \cap S_j \right)
\]

where \( g(S) = \dim(S) = \sum \alpha_i \cdot p_i \) is the weighted sum of the target points and their values, and the second terms account for the points of \( S \) also claimed by some other agent. In this example, all the locations are given the same values \( \omega = 1 \), so the offer update removes from the set the furthest one when the desired share reaches the next integer value. In this case, the termination rule can be simplified: discount factors are not taken into account, and one agent accepts an offer if its current share does not intersect with other agents’ shares, taking the exit option, while the others can continue the negotiation to resolve local conflicts (claimed locations). Since negotiation ends with no intersections, the final index \( G \) gets the optimal value of \( G = N \) in all runs. In this series of tests, the number of target points was \( n = 2 \ldots 7, k = 2n, \mathbb{P} = \mathbb{R}^k \) and \( x = \{ p_1, \ldots , p_k \} = \{(x_1,y_1), \ldots , (x_n,y_n)\} \).

Tables 3 and 4 report the results of the simulations. Each entry in the tables represents the percentage of the distance travelled by the robots in excess with regard to the optimum distance (average over 10 runs), calculated with a complete enumerative \( A^* \) algorithm, in terms of both the total distance travelled by the robots and the maximum distance travelled individually by all the robots. The \( A^* \) recursively builds a tree with all possible combinations of robots/target points, “cutting” branches where the cost of the actual partial solution (the subset of robots/target points matches) plus the estimated cost of the remaining part to have a full solution (full assignment of robots/targets) is higher than the cost of the best solution found so far.

The scenario is composed by \( N \) points randomly positioned in a square of a normalized size of \( 1 \times 1 \), while the robots were positioned at the middle of the bottom of the square.

As for the total distance travelled, the distributed assignment (negotiated solution) achieved values close to the
optimum in a few cases. In most cases, the difference is within the range 5-30%, and in a few cases it is over 100% (Table 3). However, it is important to compare the maximum individual distance travelled by the robots. In fact, the optimal solution may imply that one of a few robots do most of the work, especially when the number of targets is small and dependent upon the locations of the target points, while the negotiation algorithm tends to distribute the work. Table 4 shows how the maximum individual distance travelled improves in the case of the distributed solution. This means that while the total work performed by the robots is not optimal, this is more evenly distributed: assigning all the target points to a single robot may be better in terms of the total distance travelled, but a bad solution in terms of the parallelism of the execution. Note that the maximum individual distance travelled relates to the total mission time; hence, the distributed solution is better in applications where time is a critical factor.

It is interesting to observe that increasing the dimensionality of the task (the number of target points) does not worsen the results, while increasing the number of agents does. This result is common to all the experiments performed.

<table>
<thead>
<tr>
<th>N</th>
<th>R</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.5</td>
<td>3.5</td>
<td>11.3</td>
<td>1.6</td>
<td>4.3</td>
<td>24.1</td>
<td></td>
</tr>
<tr>
<td>130.2</td>
<td>36.4</td>
<td>12.4</td>
<td>10.1</td>
<td>18.2</td>
<td>13.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>102.6</td>
<td>51.5</td>
<td>17.6</td>
<td>21.1</td>
<td>16.8</td>
<td>6.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>137.0</td>
<td>99.4</td>
<td>117.2</td>
<td>33.9</td>
<td>58.6</td>
<td>29.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>198.6</td>
<td>141.7</td>
<td>59.3</td>
<td>36.7</td>
<td>46.4</td>
<td>34.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56.2</td>
<td>144.7</td>
<td>77.2</td>
<td>85.2</td>
<td>32.3</td>
<td>32.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Comparison of the total distance travelled with regard to the optimum solution (percentage of additional distance). Each entry is the average over 10 runs.

<table>
<thead>
<tr>
<th>N</th>
<th>R</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-20.8</td>
<td>-45.7</td>
<td>-59.9</td>
<td>-56.6</td>
<td>-65.9</td>
<td>-74.2</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>-35.7</td>
<td>-46.8</td>
<td>-56.6</td>
<td>-58.6</td>
<td>-61.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-36.5</td>
<td>-37.9</td>
<td>-49.7</td>
<td>-54.0</td>
<td>-57.7</td>
<td>-63.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-27.9</td>
<td>-49.6</td>
<td>-42.0</td>
<td>-55.6</td>
<td>-53.8</td>
<td>-55.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-22.4</td>
<td>-31.4</td>
<td>-56.6</td>
<td>-55.9</td>
<td>-61.6</td>
<td>-60.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1.3</td>
<td>-32.9</td>
<td>-51.2</td>
<td>-58.5</td>
<td>-57.5</td>
<td>-60.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Comparison of the maximum individual distances travelled with regard to the optimum solution (percentage of additional distance). Each entry is the average over 10 runs.

5. Concrete applications

In this section, we present three concrete instantiations of our task partitioning and assignment.

The first one focuses on exploration, with an application to search and rescue (SAR) [21]. The second one is an area partitioning application for aerial mapping in precision agriculture [4]. Finally, we present an example of how the negotiation strategy can be extended to tight cooperation applications - in this case box bushing - by periodic renegotiations [3].

5.1 Exploration

In this application, the mission is to explore and map a disaster scene in semi-autonomous mode. An operator at the command and control centre receives information concerning the robots’ positions and status, partial maps and other sensory information, and has to analyse such information in order to guide the search process. One of his/her duties is to decide as to which locations of the scenario to send or drive the robots for further exploration, whether it be mapping an unexplored part or revising an interesting feature (e.g., a possible victim).

In this system, the human-robot interface consists of a base station that allows the operator to monitor the progress of the mission, to know the status of the robots, give directives to them and even tele-operate them [30]. In semi-autonomous operation mode, a human operator specifies a number of interesting locations that the robots will visit and the robots autonomously navigate to them. In this mode, the operator has to specify which robots goes to which points. In the case where the number of robots comprising the team grows, it becomes increasingly difficult for the operator to supervise them.

In order to aid the operator involved in such a process and reduce his/her workload, it is useful to consider the team as a whole and assign the set of objective points to the team. Once the team has received the task, it can self-organize in order to decide which is the most suitable robot for each of the points, taking into account the robots’ preferences and limitations with regard to the actual context.

The robot coordination mechanism proposed in this work provides the operator with such a view of the team as a whole: the operator decides upon the locations to be visited (the task) and lets the team organize itself in order to accomplish the task, thus reducing the operator’s workload. Figure 5 shows two stages of a test. The squares in the centre represent the robots, while the crosses are the points which the operator would like to be visited in order to enlarge the explored area and to check for a possible victim.

Thus, in this case the agents had to negotiate how to partition a given set of target locations. The task is then the

---

7 This application has been carried out in cooperation with the Intelligent Control Research Group of the CAR-UPM in the context of the RoboCup rescue competition (www.robocuprescue.org)
set of coordinates of the points. The task evaluation function used is:

\[ g_i(S_i) = \text{dim}(S_i) - \text{dim}(\bigcup_{j \neq i} (S_i \cap S_j)) - p(S_i) \]

where \( \text{dim}(S_i) = |S_i| \) is the number of points to visit, the second term accounts for the points of \( S_i \), also claimed by some other agent, and \( p(S_j) \) is the shortest path connecting the points, \( S_j \), computed with an \( A^* \) search algorithm.

![Figure 5. Allocation of target points in robot search and rescue: initial (left) and final (right) configuration. The operator indicates the locations to be visited by the team (crosses of the left image) while the robots autonomously decide a subdivision. The image on the right shows the robots' final positions and the path travelled. Note how the map is updated as a consequence of the exploration. Note also the curved paths due to the initial robots' orientations.](image)

All the other settings were the same as those used for the numerical simulation (Sec. 4.2), with the exception that the offer update removes from the set the one that reduces by the most the total path the robot has to travel, optimizing term \( p(S_i) \).

The same experiment has been reproduced deploying the negotiator agents on a fleet of mobile robots (WiFiBots 4G, equipped with an electronic compass and GPS for self-localisation) running suitable navigation software.

![Figure 6. Snapshots of the tests with real robots. Different solutions are agreed according to the impatience of the negotiators. In this example, \( \delta_1 = 0.887 \), \( \delta_2 = 0.877 \) and \( \delta_3 = 0.867 \) (left) and \( \delta_1 = 0.887 \), \( \delta_2 = 0.867 \) and \( \delta_3 = 0.877 \) (right). The green cones represent target locations and the superimposed lines represent the final allocation.](image)

The test scenario was composed of three robots and five target points, located in an area of 15x15 metres. Figure 6 shows two different allocations, depending on the different parameters' settings (the discount factor of the robots). In this experiment, the optimum solution considering the total length of the travelled path is of 15.58 metres, while the total travelled distance of the negotiated allocation was 17.87 (left) and 18.01 metres (right), approximately 15% longer than the optimum.

This experiment also shows how the sub-task allocation depends on the discount factors of the robots. In fact, different discount factors lead to different expected values, as described earlier. Agents accept solutions that are worst
for them if they are more impatient. In the example shown in Figure 6, the two rightmost robots (robot R2 and robot R3) agree to leave point P4 to a team-mate according to their discount factor: the one with the smaller discount factor (i.e., the most impatient) gives up the negotiation earlier, receiving a smaller share of the task.

5.2 Aerial survey

Here, we applied our negotiation system to one application of robotics-based precision agriculture that requires the detection and localization of weed (undesired wild plants) in crop fields to be properly treated (e.g., by the localized diffusion of chemical agents) by ground robots. The mission described here is to take a set of geo-referenced aerial images of a given area. These images will be analysed with image-processing tools in order to detect and localize weeds. From the task subdivision point of view, this application is an area partitioning instance like those described in Sec. 4.1.

In our experiments, a team of three unmanned aerial vehicles (UAVs) is used for this purpose, two Hummingbird quad-rotor platforms\(^8\) and one AR100 platform\(^9\). Due to processing capabilities restrictions, the negotiating agents and the path planning algorithm run on a base station. Figure 7 (left) shows the architecture of the system and snapshots of the UAVs during tests at the CAR-UPM premises. Figure 7 (centre) illustrates a vineyard area nearby Madrid, Spain, that has been a theatre of the field tests, and a solution of the area partitioning. The individual flight plans of the three agents is shown on the right.

![Figure 7: Left: architecture of the aerial robots’ team and snapshots of the UAVs in action. Centre: vineyard field objective of the mission. The field has an irregular shape of approximately 330 × 200 metres. Area partition is done in the continuous space. Right: flight routes in the discrete space (red lines). For the purpose of path planning, the area is discretized in cells (10 × 10 cells of size 32.7 × 20 metres), where the centres of the cells are waypoints for the UAVs and the cells correspond to one picture of the global mosaic.](image)

The experiments performed have shown the usefulness of the autonomous negotiation process: the operator simply specifies the area to be surveyed, with a set of geo-referenced points, and the number (and kinds) of vehicles available. The negotiator agents will produce a candidate area partitioning and the flight plans are calculated. The operator can review and validate the solution and, upon validation, flight plans are sent to the UAVs for the mission to start. For more details of the process and experimental results, we refer the reader to [4].

5.3 Box pushing

In order to explore the potential and limitations of the proposed approach, we present a final instantiation of our method. Box-pushing is a classical test bed application for high cooperation strategies, where agents have to coordinate their actions in a continuous manner. Even if the negotiation protocol presented here targets loose cooperation problems, it can be applied to this problem by adopting a periodic renegotiation of the tasks involved.

Figure 8 (left) illustrates how the problem can be expressed in our task/sub-task formalism. Task are vectors representing the direction of the movement of the box if they are executed. Tasks are described by two parameters \(x_i=[L_i, F_i] \in \mathbb{R}^2, i=1,2\). These are interpreted as a push location and a push force by the agents (Figure 8 (right)). The execution of the robots’ tasks consists in pushing the box at the push location \(L_i\) with force \(F_i\) during \(\Delta t=1\) sec.

---

The combined execution of the sub-tasks produces a displacement and a rotation of the box towards its target location. Thus, the robots have to negotiate how each of them will push the box to obtain the desired motion \( T_0 \) described as a vector of desired motion. When robot \( j \) receives an offer \( \tau_j \), it has to compute what its push location and force \( x_j \) should be, such as where the translation and rotation of the box \( (T_j, R_j) \) produced by the combination of the two pushing actions is equal to the desired rotation and displacement of the box, i.e., \( T_0 \). If this can be found, agent \( j \) accepts the offer; otherwise, it makes a counteroffer to agent \( i \).

The task evaluation function used in this case is simply:

\[
g_i(S_i) = g(S_i) = S_i \cup S_j
\]

i.e., how much "useful" motion is obtained by the combination of the two combined pushing actions.

Figure 9 shows a snapshot of the experiments performed, using two modular mobile robots (see [2] for more details on the robots). The robots communicate with a central control station via Bluetooth wireless connections. A fixed camera located on top of the scenario is used to detect the robots’ position and orientation, as well as the position of the box, providing all the necessary information for the negotiation to take place. The negotiator agents run in the central control station. We refer the reader to [3] for a detailed description of the experimental results.

In this example, we can see how dynamic allocation using renegotiation is achieved. After the agreed actuation of the robots, the target task \( T_0 \) changes. For this reason, the subdivision has to be renegotiated after each action, which is after each \( x \) time segment. Note that the renegotiation can deal with changes in the target location, i.e., even if the mission objectives change, the system is capable of negotiating the correct division of labour in dynamic environments.

Fig. 10 illustrates an example run. On the left, a graphical representation of the bar position over time is reported, while on the right the evolution of the bar’s orientation compared to the desired orientation is reported.

Figure 8. Encoding of the box-push task in our framework. Left: the parameters describing the task are push force and push direction. Right: if \( S_1 = \tau (x_1), S_2 = \tau (x_2) \) is a given decomposition of \( T_0 \) in two sub-tasks, then the operator \( \boxdot \) is the projection of the composition parallel to \( T_0 \) (the "useful" component), and the operator \( \blacktriangleleft \) is the projection of the composition orthogonal to \( T_0 \). It is easy to see that, in the ideal case, \( S_1 \cup S_2 = T_0 \) and \( S_1 \cap S_2 = \emptyset \).

Figure 9. A snapshot of one box-pushing application. The green and yellow circles represent the agreed pushing points and the black arrow represents the desired motion vector. This experiment has been carried out in collaboration with Mr. J. Baca and Prof. M. Ferre, Group of Robots and Intelligent Machines, CAR UPM-CSIC.

Figure 10. Example run. Left: the red dotted line represents the bar and the big circles represent the positions of the robots. Right: box direction with regard to the desired direction.
6. Conclusions

The main contributions of this paper are twofold. First, we propose a formal definition of the concept of a task, general enough that allows for the expression of different problems. A task partitioning algorithm implemented using such a formulation can then be applied to a wide variety of multi-robot tasks. Second, a negotiation protocol that takes advantage of theoretical results coming from game theory which provide some important properties, such as termination criteria and the prediction of the outcome, that can be used to drive the negotiation. Moreover, the algorithm is able to be extended to incorporate mission-specific features such as exit options and the renegotiation of a solution to adapt to a task change.

Experiments conducted both in computer simulations and real applications demonstrate the effectiveness of the proposed approach, which allows for the quick instantiation of the general framework to the particular cases, i.e., an easy step from theory to practice. Here, we have presented three examples with different kinds of missions and robots, with the purpose of illustrating the versatility of our approach. Indeed, our aim was to provide an abstraction of the concepts of task, reward and negotiation, general enough to embrace a large set of possible multi-robot missions, and at the same time aimed at being actually implemented and deployed in real MRSs.

This is also possible because vehicle-specific features are hidden in the task evaluation function of the robots, and thus there is no need for the system or its team-mates to be aware of them. This makes the system more general and decouples the robots’ particular characteristics from the negotiation protocol, allowing for the use of teams of heterogeneous robots.

In conclusion, the framework we have presented provides a fully distributed, simultaneous task subdivision and allocation system for teams of heterogeneous robots that can be applied to a wide variety of different robot cooperation problems.

7. Acknowledgements

This work was partially funded by the projects ROTO (Fleet of autonomous aerial and ground robots - DPI2006-03444) of the Ministry of Education and Science of Spain, RHEA (NMP-CP-IP 245986-2) of the European Commission, and ROBCITY2030 of the Community of Madrid, Spain.

8. References

[16] Sedat NazLibeke. Autonomous multiple teams establishment for mobile sensor networks by


